strace Syscall tracer

Your program traced process → syscall → **strace** ptrace stops it, logs → resume → **Kernel** does the work

CONCEPTS

strace prints every system call a process makes: name, arguments, return value. Output goes to stderr.

Read lines as name(args) = result; failures end in -1 plus an errno such as ENOENT.

Tracing slows the target a lot. Use filters and avoid it on busy production processes.

RUN AND ATTACH

strace *<cmd>* **[args]**
Trace a new command

strace -p *<pid>*
Attach to a running process

strace -f *<cmd>*
Follow forks, clones and threads

strace -p *<pid>* **-p** *<pid2>*
Attach to several processes

strace -o *<file>* *<cmd>*
Write the trace to a file

strace -A -o *<file>* *<cmd>*
Append to the output file

RUN AND ATTACH (cont.)

strace -ff -o *<prefix>* *<cmd>*
One output file per process

strace -E *<VAR>*=*<val>* *<cmd>*
Set an environment variable for the command

strace -u *<user>* *<cmd>*
Run the command as another user

strace -b execve *<cmd>*
Detach once the program calls execve

strace --seccomp-bpf -f *<cmd>*
Stop only at traced syscalls (faster)

FILTER SYSCALLS

strace -e trace=_<name>_,_<name>_ *<cmd>*
Only these syscalls

strace -e trace=%file *<cmd>*
Syscalls taking a file name

strace -e trace=%network *<cmd>*
Network syscalls

strace -e trace=%process *<cmd>*
Fork, exec and exit

FILTER SYSCALLS (cont.)

strace -e trace=%desc *<cmd>*
File descriptor syscalls

strace -e trace=%memory *<cmd>*
Memory mapping syscalls

strace -e trace=!*<name>* *<cmd>*
Exclude a syscall

strace -P *<path>* *<cmd>*
Only calls touching this path

strace -Z *<cmd>*
Show only failed calls

OUTPUT FORMAT

strace -s *<n>* *<cmd>*
Print strings up to n bytes

strace -y *<cmd>*
Show paths for file descriptors

strace -yy *<cmd>*
Also show socket addresses and devices

strace -v *<cmd>*
Print full structures, no abbreviation

strace -e write=*<fd>* *<cmd>*
Hex dump of data written to fd

OUTPUT FORMAT (cont.)

strace -e read=*<fd>* *<cmd>*
Hex dump of data read from fd

strace -k *<cmd>*
Stack trace after each syscall

TIMING AND STATISTICS

strace -c *<cmd>*
Per-syscall summary instead of a trace

strace -C *<cmd>*
Summary plus the normal trace

strace -c -S calls *<cmd>*
Sort the summary by call count

strace -T *<cmd>*
Show time spent in each syscall

strace -tt *<cmd>*
Wall-clock time with microseconds

strace -ttt *<cmd>*
Unix epoch time with microseconds

strace -r *<cmd>*
Time since the previous syscall

TAMPERING

strace --fault=_<call>_**:error=**_<ERRNO>_ *<cmd>*
Make a syscall fail

strace --inject=_<call>_**:error=**_<ERRNO>_**:when=**_<n>_ *<cmd>*
Fail only the nth call

RECIPES

Why does it fail to start?

```
# which files does it look for, and what is missing?
strace -f -e trace=%file -Z -o trace.txt <cmd>
grep ENOENT trace.txt
```

Where is the time going?

```
strace -c -f -o summary.txt <cmd>
# then slow single calls
strace -f -tt -T -o trace.txt <cmd>
```

Debug a hung process

```
# last call shows what it waits on
sudo strace -f -tt -p <pid>
# connect/poll on fd? map it to a socket
sudo strace -yy -e trace=%network -p <pid>
```

See what config files are read

```
strace -f -e trace=openat -y -o trace.txt <cmd>
grep -v ENOENT trace.txt | grep -E 'conf|\.ini|\.yml'
```